Proyecto Final: Corrector Ortográfico en Español

Instituto Tecnológico Autónomo de México
Maestría en Ciencias en Computación
Karen L. Poblete 116452
Estadística Computacional

18 de diciembre de 2015

1. Introducción

Desde los años 90 se han diseñados diferentes modelos del lenguaje utiles para correguir la ortografía de textos. Los primeros y más importantes trabajos donde se basa este proyecto se basan en los artículos del laboratorio Bell de AT & T [1] [2] y el proyecto Watson de IBM [3]. Este método se basa en inferencia Bayesiana y modela los errores como un canal ruidoso, donde las faltas ortográficas son tratadas como si la palabra correcta pasa por un canal distorsionado por el ruido. El ruido es el causante de las faltas y el objetivo es descubrir la palabra que realmente fue pasada por el canal, que sería la palabra correcta.

El modelo de canal ruidoso se basa en la inferencia Bayesiana, ya que apartir de x, siendo la palabra erronea, se puede llegar a w que generó la palabra x. Siendo V el conjunto de las palabras disponibles en el lenguaje, el vocabulario, queremos encontrar w tal que la probabilidad $P(w|x)$ es la más alta [4].

$$\hat{w} = \text{argmáx}_{w \in V} P(w|x)$$

Utilizando intuición Baayesiana se puede llegar a:

$$\hat{w} = \text{argmáx}_{w \in V} \frac{P(w|x)P(w)}{P(x)}$$

Donde $P(x)$ se puede dejar fuera, ya que no cambia mucho de palabra a palabra, porque los errores cometidos tienen una probabilidad similar. Es decir, la mayor parte de los errores que se cometen están relacionados con omitir, agregar, trasponer o cambiar una letra y la probabilidad de tener sólo un cambio es el mismo; el 80 % de las faltas ortográficas son de est tipo y el 20 % tienen más de un cambio. Dado ésta característica podemos entonces calcular mejor:

$$\hat{w} = \text{argmáx}_{w \in V} P(w|x)P(w)$$

Donde $P(w|x)$ depende del modelo del canal, es decir, el modelo de los errores ortográficos y $P(w)$ del modelo del lenguaje, el cual está basado en la frecuencia de las palabras. El modelo de lenguaje se puede generar por medio de n-gramas, es decir, hacer un análisis de la frecuencia de palabras individuales, unigramas. También se puede utilizar modelos de bigramas y trigramas, donde las probabilidades de aparición de una palabra depende de las n palabras anteriores. En el caso de bigramas, la aparición de la palabra actual depende de la que está justo antes y para los trigramas de las dos que la anteceden. Los correctores léxicos basados en trigramas han mostrado mejores resultados cuando las palabras erroneas existen en el vocabulario y dependen de contexto [4]. El contexto puede estar representado por Cadenas de Markov, donde el estado actual, que sería la palabra en error depende de la palabra pasada. Este proyecto se relaciona con la clase ya que utilizo metodos de inferencia bayesiana y cadenas de Markov para el contexto, donde se aplica la regla de que el estado actual sólo depende del anterior.

2. Candidatos de palabras correctas

Modelando el sistema de corrección de palabras como se ha dicho con anterioridad, se requiere encontrar los candidatos que puedan maximizar la función decrita con anterioridad. Para esto es necesario generar dichos candidatos haciendo eliminaciones, inserciones, sustituciones y trasposiciones de letras en la palabra erronea. Eliminación se refiere a quitar una letra, sistitución a cambiar una letra por otra, inserción a agregar una letra y trasposición a cambiar de lugar una letra con otra en lugares contiguos.

Existe una medida de distancia entre palabras conocida como distancia de Damerau-Levenshtein la cual indica el número de modificaciones que se realizan a una palabra para generar otra. Por ejemplo, de "casa" a "caza" la distancia es 1, ya que sólo cambió una letra. En el caso de "casa" y "acza" sería 2, ya que hay una trasposición y una susutitución. Como se dijo con anterioridad, los errores más comunes son de distancia 1, por eso los candidatos que mayormente se generan se encuentran a esta distancia, seguida de los de distancia 2.

3. Modelos de lenguaje

Como se ha mencionado con anterioridad existen diferentes maneras de modelar el lenguaje. Cuando se requiere de un modelo de baja complejidad, se pueden utulizar unigramas. Este modelo de lenguaje no puede representar ningún tipo de relación entre la sucesión de las palabras, por lo que se considera ausente de contexto. Con este tipo de modelo de lenguaje se pueden resolverciertos casos de errores ortográficos donde la palabra erronea no existe en el vocabulario, por lo que es sencillo encontrarla.

Cuando se quere abordar errores relacionados no sólo con palabras carentes de significado y presencia en el vocabulario, sino también todas esas palabras que fonéticamente suenan igual pero tienen significados diferentes y por lo tanto su error depende del contexto en el que se utilizan. En este caso se calcula la frecuencia de los bigramas y se calculan sus probabilidades.

También es posible utilizar trigramas, es decir, la aparición de una palabra depende de las dos anteriores. Se ha comprobado que los modelos de lenguajes con trigramas muestran mejores resultados que con bigramas y unigramas. Lo negativo es la cantidad de información que se tiene que utilizar para tener trigramas significativos, además de los recursos de computación. Calcular trigramas es más complicado y complejo que unigramas y bigramas.

Una frase es una sucesión de palabras que se encuentran en el vocabulario $v = w_1, w_2, ..., w_n$. Entonces una frase se puede representar como:

$$W = w_1 w_2 w_3 w_4 ... w_n$$

Necesitamos calcular la probabilidad del error x en la frase W como $P(W|x)$ y la probabilidad de cada frase $P(W) = P(w_1...w_n)$. Esto nos lleva al uso de la regla de la cadena, relacionada directamente con cadenas de Markov.

$$P(w_1...w_n) = P(w_1)P(w_2|w1)P(w_3|w_1w_2)...P(w_n|w_1...w_{n_1})$$

Basta calcular la palabra que aparece después de una sucesión:

$$P(w|w_1...w_m)$$

En el caso de bigramas, se puede simplificar:

$$P(w_1...w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})$$

Y con trigramas:

$$P(w_1...w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_{n_2}w_{n-1})$$

La versión más sencilla es con unigramas:

$$P(w_1...w_n) = P(w_1)P(w_2)...P(w_n)$$

Lo malo de los modelos con n-gramas es lo ralo de los datos, es decir, se requiere un corpus muy grande para que se pueda cubrir una cantidad buena de casos. Para muchos pares de palabras, no habrá información suficient en el set, por lo qe se requiere suavizar las probablidades. Es decir, asignar una probabilidad a las parejas que no existen en la base. El método más sencillo es utilizar suavizamiento de Laplace.

$$p^*(w|z) = \frac{c(zw)+1}{c(z)+V}$$

3.1. Para evaluar los modelos de lenguaje

Aunque los modelos de lenguaje son mejor evaluados en el contexto de su aplicación, esto generalmente es muy costoso. Podemos hacer sin embargo evaluaciones simples como en modelos de predicción. Consideramos entonces un modelo que ajustamos con un conjunto de entrenamiento que nos da probabilidades $\hat{P}(w_1 \cdots w_m)$ para frases $w_1 \cdots w_n$. La idea es tomar un conjunto de prueba (o validación) y estimar la verosimilitud de las frases de entrenamiento. Mejores modelos dan verosimilitud más alta.

Entonces, suponiendo que $W = w_1 \cdots w_n$ son nuestros datos de prueba (separando diferentes oraciones con $< s >$ y $< /s >$ por ejemplo), podríamos calcular la verosimilitud:

$$L = \prod_i \hat{P}(w_1 \cdots w_n).$$

En el modelo de bigramas, esto se reduce a

$$L = \prod_i \hat{P}(w_i|w_{i-1})$$

Invertimos y normalizamos por el número de palabras para obtener la "perplejidad":

$$PP = \left(\prod_i \frac{1}{P(w_i|w_{i-1})} \right)^{1/n}$$

, y buscamos "minimizar" la perplejidad.

Nótese que la devianza $D = -\frac{2}{n} \log L$ satisface $2 \log PP = D$, de modo que se trata de minimizar la devianza

$$D = -\frac{2}{n} \sum_i \log P(w_i|w_{i-1}).$$

Cuando se utilizan bigramas, se da por hecho que cada palabra tiene alguna probabilidad de ser erronea o no. Por eso se utiliza la siguiente formula para calcular dichas probabilidad, donde x es el error observado y w la palabra correcta.

$$p(x|w) = \begin{cases} \alpha & \text{if } x = w \\ \frac{1-\alpha}{|C(x)|} & \text{if } x \in C(x) \\ 0 & \text{otro caso} \end{cases}$$

4. Modelos de errores y ruído en el canal

El desepeño del algoritmo no solo depende de el modelo de lenguaje, sino también del modelo del canal ruidoso, o de los errores. Se debe tener una referencia de los errores. Muchos de los errores ortográficos que se cometen tienen relación con la fonética de las palabras, por ejemplo: "decisión" por "desisión", donde la c y la s tienen el mismo sonido. Para generar el modelo de errores se puede crear una matriz de confusión de las sustituciones que se realizan con las letras. En este caso se requiere de un set con los errores más frecuentes.

Wikipedia ofrece una liste de los mismos para diferentes idiomas. De la matriz de confusión se calculan las probabilidades de cometer cada error. Las probabilidades se pueden modelar de diferentes maneras, como errores libres de contexto y sólo relacionados con la letra que se modifica o dependientes de contexto, es decir, que están relacionados con las letras que lo anteceden y proceden dentro de la palabra.

Otra manera de modelarlo muy eficiente es establecer que toda letre tiene una probabilidad de estar erronea dentro de una palabra y asignar probabilidades a cada una de ellas. Las probabilidades serían similares, pero se pueden diferenciar según la distancia a la palabra correcta. A mayor distancia menos probable es que sea erronea.

Durante el experimento, se consiguió una muestra de los errores más comunes de Wikipedia en español. La muestra tiene 430 errores, con los cuales se probó el sistema. Se evaluó la distancia entre la palabra correcta y el resultado del sistema para ambos correctores, es decir el el creado con unigramas y bigramas. Ambos mostraron tener desempeños similares. El shiny muestra dichos errores, en una pestaña y los resultados obtenidos.

5. Objetivo

El objetivo del presente proyecto es generar una aplicación basada en Shiny the R, que utilice el anterior modelo descrito para corregir texto. Se crearon dos versiones, una versión modelando el lenguaje por medio de unigramas y el segundo utilizando bigramas. La aplicación elaborada con Shiny también muestra a detalle el proceso de corregir las palabras.

6. Algoritmos y métodos

En ésta sección se relata el proceso que se llevó a cabo para generar los dos correctores ortográficos generados aquí. En ambos casos se utilizó un set de entrenamiento con 50,000 notas de periódico en español. Dichas notas fueron limpiadas de tal manera que todas las letras fueran minúsculas y los diferentes signos de puntuación y demás signos no alfa numéricos fueron removidos. Con este set de entrenamiento se contaron las frecuencias de cada unigrama y bigrama, después se calculó de acuerdo al número total de los mismos, la probabilidad de aparición en escala logarítmica.

Una vez contando con el los valores de frecuencias de los unigramas y bigramas se realizan los siguientes pasos. Todas las funciones de este proyecto fueron desarrolladas en R. En el caso de utilizar unigramas:

1. Encuentran las palabras erroneas en el texto de entrada.
2. Por cada palabra erronea se generan candidatos por medio de las cuatro operacion básicas mencionadas con anterioridad: eliminación, inserción, sustitución y trasposición.
3. De los posibles candidatos sólo quedarse con los que se encuentran en el vocabulario y desechar las demás.
4. Calcular la probabilidad del modelo de canal de ruido para cada candidato en escala logarítmica.
5. Sumar la probabilidad de aparición de dicha palabra con la probabilidad proveniente del modelo del canal.
6. Nos quedamos con la palabra que tenga una probabilidad mayor.

El modelo de unigramas es un modelo sencillo en la parte del modelo de lenguaje, donde sólo se considera la frecuencia de ocurrencia de las palabras. El modelo tiene la ventaja de ser rápido y con complejidad moderada, pero no tiene un buen desempeño cuando los errores son palabras que existen en el diccionario. Para resolver estos casos es necesario contar con contexto.

El modelo de lenguaje que utiliza bigramas se basa en la probabilidad condicional de que una palabra aparezca seguida de otra. Ésta es una manera de tener contexto ya que cada palabra depende de la anterior.

Para que este algoritmo funcione es necesario contar con una probabilidad para cada palabra de que sea erronea ya que queremos atacar errores donde las palabras existen en el diccionario pero no van de acuerdo al contexto. Algunos ejemplos de estos casos, sería cuando en lugar de "casa" se escribe "caza". Ambas palabras existen en el diccionario pero se puede saber si son erroneas de acuerdo al contexto. El algoritmo de este modelo es:

1. Encuentran las palabras erroneas en el texto de entrada.
2. Por cada palabra se generan candidatos por medio de las cuatro operacion básicas mencionadas con anterioridad: eliminación, inserción, sustitución y trasposición.
3. De los posibles candidatos sólo quedarse con los que se encuentran en el vocabulario y desechar las demás.
4. Por cada par de candidato y la palabra que antecede al error x calcular la frecuencia de su bigrama.
5. Por cada par de candidato y la palabra que le procede al error x calcular la frecuencia de su bigrama.
6. Multiplicar las frecuencias de ambos bigramas.
7. Calcular la probabilidad del modelo de canal de ruido para cada candidato en escala logarítmica.
8. Sumar la probabilidad de aparición de dicha palabra de acuerdo a los bigramas con la probabilidad proveniente del modelo del canal.
9. Nos quedamos con la palabra que tenga una probabilidad mayor.

En este modelo de lenguaje con bigramas se condiciona la aparición de la palabra que genera el error x a la palabra que le antecede y que le procede, esto ayuda a revisar las palabras por contexto. También es necesario agregar que se checan todas las palabras como casos posible ser erroneas, por que como se ha explicado con anterioridad, todas las palabras son sospechosas de error ya que por contexto podrían estar mal.

7. Resultados

Adjunto a este reporte se presenta la aplicación desarrollada en Shiny. Ambos correctores fueron programados y se pueden utilizar. El programa con unigramas resultó tener un mejor desempeño con las palabras que están erroneas que no son parte del diccionario, ya que es más fácil detectarlas. En el caso del programa con bigramas, el desemeño depende del corpus utilizado en el entrenamiento. Mientras memor sea el set de entrenamiento el desempeño baja. Se requiere una gran cantidad de información para hacerlo funcionar bien. En este experimento no se cuenta con los recursos computacionales para procesar mas de 20,000 textos para generar bigramas y trigramas. Con 20,000 se pueden tardar varias horas en procesar el corpus, por lo que los experimentos que se realizaron con el sistema utilizaron corpus pequeños.

8. Conclusiones

Utilizar un modelo del lenguaje con unigramas puede resolver problemas ortográficos donde las palabras no cuentan con contexto, puede considerarse una primera aproximación a los correctores ortográficos. El modelo es sencillo y rápido pero no puede detectar errores que caben en el vocabulario pero están mal empleadas, como son los errores fonéticos que dependen del sonido.

El modelo de lenguaje con bigramas es un poco más complejo pero puede ser útil para corregir palabras por contexto. El modelo se basa en la regla de Markov, donde el estado actual depende sólo del anterior y de inferencia bayesiana.

Referencias

[1] Kernighan, M. D., Church, K.W., and Gale,W. A. *A spelling correction program base on a noisy channel model*. In COLING-90, Helsinki, Vol. II, pp. 205–211, 1090.

[2] Church, K. W. and Gale, W. A. *Probability scoring for spelling correction. Statistics and Computing*. 93–103, 1991.

[3] Mays, E., Damerau, F. J., and Mercer, R. L. *Context based spelling correction. Information Processing and Management*.517–522, 1990.

[4] Jurafsky, D., and Martin J.. *Speech and Language Processing*. Chapter 6. 2014

[5] Norvig, P. *How to write a spelling corrector*. www.norvig.com/spell-correct.html. 2007.

[6] Norvig, P. *Natural language corpus data*. In Segaran, T. and Hammerbacher, J. (Eds.), Beautiful data: the stories behind elegant data solutions. O'Reilly. 2009.

[7] Colins, M. *Language Modeling*. Course Notes for COMS w4705.